UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

ENGINEERING FACULTY

Title: Security in 1-wire system. Case study: Home automation

Author: Luz Alejandra Magre Colorado

Jury

Jury

Supervisor:

Cartagena de Indias, December 2017

Security in 1-wire system. Case study: Home automation

Luz Alejandra Magre Colorado
Supervisor: Juan Carlos Martínez Santos

Universidad Tecnológica de Bolívar
Faculty of Engineering
Master in Engineering with Emphasis in Electronics
Cartagena de Indias, D. T. y C.

December 2017

Security in 1-wire system. Case study: Home automation

Luz Alejandra Magre Colorado

Trabajo de grado para optar al título de

Máster en Ingeniería con Énfasis en Electrónica

Director: Juan Carlos Martínez Santos

Universidad Tecnológica de Bolívar
Facultad de Ingenierías
Cartagena de Indias, D. T. y C.

Diciembre 2017

Resumen

La automatización de viviendas es un campo de la tecnología que siempre se encuentra en crecimiento, desarrollando sistemas que reducen los costos de los dispositivos. Por esto, se ha logrado que la domótica esté al alcance de todos. Desde la aparición de productos que permiten crear tu propio sistema domótico, y la reciente popularidad que ha tenido el Internet de las cosas (IoT), la industria de la automatización de viviendas ha cambiado mucho. Tener la habilidad de controlar dispositivos a través de Internet crea numerosas vulnerabilidades al sistema, permitiend a un atacante controlar y ver todo lo que ocurre. En este trabajo se estudia un sistema domótico que usa 1-wire como protocolo de comunicación. Originalmente, el sistema carece de seguridad. Nuestro objetivo es implementar seguridad de la información a través de la encriptación de los comandos del sistema, para así poder proveer Confidencialidad, Integridad y Disponibilidad (CIA). Los resultados muestran no sólo la implementación exitosa del módulo criptográfico dentro del sistema domótico para proveer seguridad, sino que también se demuestra que añadir este proceso no afectaría el modo en que el usuario maneja sus dispositivos.

Palabras clave: Domótica, Seguridad, 1-wire.

Abstract

Home automation is a technology that is permanently growing thanks to te development of devices that reduces the costs of these kind of systems. This makes domotics an affordable option to everyone.

Thanks to the appearance of this low cost devices, and the recent popularity around the Internet of Things (IoT) technology, the domotics industry has changed to a new level. Having ability to control devices through the Internet creates numerous vulnerabilities to an unprotected system, allowing an attacker to control or see everything. In this work, we present a domotic system which uses the 1-wire communication protocol. Originally, the system lacks of any kind of security. Our objective is to implement information security through the encryption of the system's commands, so we can assure the CIA triad (Confidentiality, Integrity and Availability). The results show not only the successful implementation of the cryptographic modulo into the system to provide security, but we also prove that adding this extra process would not affect the way the final users handle and use their devices.

Keywords: Domotics, Security, 1-wire.

Special Thanks

First and most important of all, thanks to my family for always supporting me in every step I take towards my goals. To my father Alexandre, who has always been my role model both as an engineer and as a person. Thank you for promoting in me the curiosity on science and teach me to always go a step further in all aspects of my life. To my mother Adela, for being the mother who is always right and teaching me that being a good professional is not always about knowing the numbers. Thank you for your words of wisdom when I needed them the most and help me see what to do when I'm stressed out. To my older sister Montse, for being my best friend and sidekick. Thank you for teaching me to relax and see the bright side on everything. You are one of the most special people in my life. To my little brother Alexandre for being unique and give me more points of view to think of that I have not considered before. To my grandparents Santiago and Delia for always taking interest on what I do and loving me unconditionally.

Thanks to the Center of Excellence and Appropriation of Internet of Things (CEA-IoT) for the economical and academic support that allowed me to be on this master's program.

Thanks to my professor Juan Carlos Martínez, for guiding me on the whole process of this master. For all his patience since my bachelor's thesis work. Your teaching has been fundamental in my formation as an engineer and researcher. Thank you for giving me the opportunity and the means to do this master's degree.

To all the friends that accompanied me on the way. Thanks to Camilo Bayter for always helping me in anyway he could. For your patience and always know whether I needed space or your company. Thanks to my "niños" in the office, Jorge Franco, Carlos Buelvas, Iván Baños, Juan S. Mantilla and Hernando Ariza. For your help when I had "coding" problems and teach me that we will always be *Teamcolombia*.

Last, but not least I give the most special thanks to God, for blessing me with health and lots of love. For all this people that came into my life and make me the person I am today.

Contents

List of Figures

List of Tables

Chapter 1.
Introduction

One of the most important considerations that a home automation system must take into account, is how to establish communication between the different devices that are connected and controlled by the smart home. Depending on how it is established, it is possible to also connect the system to the Internet, allowing the user the possibility of controlling everything remotely. This new ability can create more disadvantages than advantages itself if there is no security implemented. An attacker could compromise the integrity and confidentiality of the system without the owner realizing that something is happening until it's too late [58, 35, 55, 51, 34]. No matter what is the real intention of the intruder, in any case it becomes a direct attack to the privacy of the inhabitants of the home.

This is enough to know that we need to ensure that the communication of the system is being established in a secure way. This means, to avoid the leaking of information and protect the data of any interference or corruption [71, 27]. We are studying a domotic system that uses 1-wire as its communication protocol, the problem is, that 1-wire is completely vulnerable to an intruder's attack. On this system, the plaintext of the commands are sent through the whole system, allowing any user

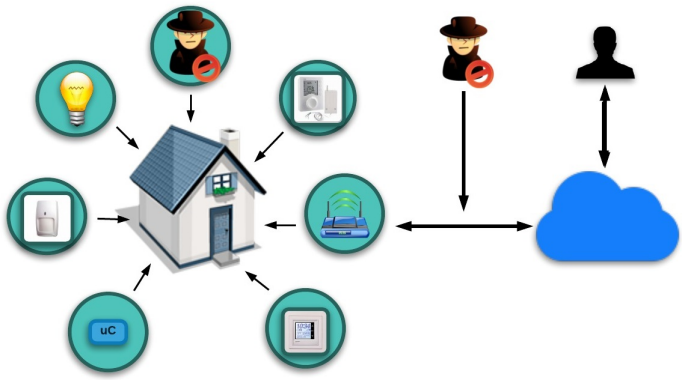

Figure 1.1: Insecure scenario of a domotic system connected to the Internet.

(authorized or not) to read, intercept (eavesdropping [70]) and change information. Plus, it is difficult to identify if something has been done without authorization. Given that there are many installations already in place using the 1-wire protocol, we are focusing on this protocol specifically.

1.1 Justification

According to Statista [74], in 2016 there were around 14.2 million smart homes in the United States alone, and this number is expected to grow to 36.01 million for 2020. With this number permanently increasing, we can't take security for granted. Domotic systems are often vulnerable to confidentiality and integrity attacks. An attacker could control a household remotely o simply look into the routine of the people living there [35]. This situation can become easily in a physical attack to the user given that the intruder could open the door remotely, or control the devices in a way that causes some sort of physical damage to the inhabitants of the house [55, 51, 34]. This is why we must prevent these attacks through the implementation

of information security in these domotic systems.

1.2 Objectives

Having in mind the problem stated above, we have set the objectives for this thesis work.

1.2.1 General Objective

To identify a cryptographic technique capable of defending itself against the more common attacks (ciphertext-only attack, known plaintext attack, chosen plaintext attack), that fulfills the security requirements (Confidentiality, Integrity, Availability) and adjusts to the 1-wire system's restrictions applied to domotics.

1.2.2 Specific Objectives

- To identify the communication restrictions and the inherent characteristics of the system.
- To select cryptographic techniques suitable for the system that can be implemented through the 1-wire protocol.
- To compare the implementation of the selected cryptographic techniques on the system to determine that the security requirements are fulfilled.

Chapter 2.
Theoretical Framework

In this section, we will talk about domotics in general, the different communication protocols used for home automation systems and their vulnerabilities on the security matter. Here we also make a revision of the restrictions that a system using 1-wire could have. This way we are fulfilling the first specific objective by describing the different limitations that we must take into account to be able to choose an strategy to solve the information security issue.

2.1 Domotics or Home Automation Systems

Domotics is defined in many ways, depending on the use and what is the user is controlling. A broad definition of home automation is "the capacity to automate and control multiple systems" [17, 45]. Others like Goodwin [31] take for instance, the objective of the automation itself, defining this kind of automation as anything the home does to make people's lives more productive or comfortable. There is also the concept that domotics is a set of systems that allow to control and automate a household through a master that manages the actions of said systems [38]. We could spend a lot of time trying to get a universal definition of what home automation is

and what represents in people's lives. But something we can agree on, is that the main goal of these systems is to automate devices that make rutinary tasks a little it easier.

2.2 History of Home Automation

Home automation systems have been around for a long time, more than you would have thought. The concept of an automated home has been developing since the creation of the first home appliances.

1901-1920: Home Appliances

The invention of home appliances was the first step to lead us to the automated home. These machines allowed housewives around the world to save time, money and energy in every day tasks. The invention of the first engine-powered vacuum cleaner (1901) started a time when electricity could help create electricity-powered machines [78]. During these two decades, besides vacuum cleaners, refrigerators, washing machines, toasters, irons, among other things were invented [32].

1966: ECHO IV

The ECHO IV was invented by James Sutherland from Westinghouse, it also was known as the first automation machine, although it wasn't a commercial success at the time. This device could performed tasks simple to us nowadays, such as control the temperature of the household, turn on and off the appliances available, make shopping lists, and even manage the family accounting [32, 64, 72].

Figure 2.1: ECHO IV machine. Known as the first home automation machine. Taken from [64].

1975: X10

X10 was the first general purpose network technology for use in home automation [9], and it's still a common communication protocol to use. It is an affordable and simple communication standard that uses the power line to remotely control the devices in the system [26, 16]. It is known to be very easy to use, install, and it does not need a central unit to control the devices [36].

1991: Gerontechnology

Home automation started to get into this combination between technology inclusion and the elder in need of assistance in the 90's. Different systems to assist the elder appeared to give a better care and be warned if something wasn't right with the

health of the person [13, 14, 29]. At this time, home automation began to be not just a luxury or something to be a little more comfortable or organized with, the development of new systems could help the elder and the handicapped to live an easier and more independent life.

Early 2000s: Smart homes

Although the concept of home automation and the longing of your own home to help you with basic tasks has been around for a long time, it's in the early 2000s when the popularity of the smart homes increased more than any other time. With the development of different technologies, it also started to become more affordable for all people [32].

2010s: Smart Homes and IoT

Nowadays, home automation is not only about automatize processes, it is about connectivity too. The importance of controlling remotely the systems is key for certain processes and to monitor the state of your household. With the new paradigm of the Internet of Things (IoT), it is possible to connect literally any device to the Internet. These systems are able to make decisions based on the data collected by the sensors installed. In today's home automation technology, we have infinite possibilities to integrate different technologies that can allow us to take the most of it.

2.3 Security Vulnerabilities on Communication Protocols

There are many ways to communicate your devices and convert your home in a smart one. The communication protocol can be wired or wireless, it all depends on the user's preference or the manufacturer's. In this section we will talk about the vulnerabilities and some countermeasures that some of the most used communication protocols in domotics have. Then, we will concentrate on the vulnerabilities of the 1-wire protocol, given that this protocol is used by the domotic system that we are studying.

2.3.1 X10

As said before, the X10 was the first network technology used for home automation, using the power line to control different devices. In an X10-based home network we can find a limited number of change state commands, these are: On, off, dim, bright, all lights on, all lights off, and all units off, extended code, hail request, hail acknowledge, pre-set dim, status on, status off, status request. The commands are all sent by a 4-bit code. Also, there is a limit of 256 devices in an X10 system [16, 39].

In this protocol, is pretty easy to perform a brute force attack on the system, given that each network has a unique 4-bit ID number. This means that there is only 16 possible ID numbers [63]. Note that there is no encryption at all, and the possibilities of interference are high.

Another flaw is that that you can turn off every device with a single command. For example, if someone connects an X10 device outside your house (if you have an electric socket outside of course), that person can turn off all of your devices. A thief can use this to turn off your alarm and get into your house [63]. Moreover, Kennedy and Simon [39], developed a pair of tools named X10 Sniffer, and X10 Blackout. With the first one, the attacker can know what the devices are doing inside the household. With the second tool, the attacker can create interference with the operation of the devices.

A simple solution seems to be the use of an isolation transformer to separate lines for x10 devices and devices that are not X10 [63].

2.3.2 Z-Wave

Z-wave (Sigma Designs, California) is a wireless proprietary protocol and RF-based [36] that allows a two-way communication. A Z-Wave network can have up to 232 devices, but the possibility of bridging two networks exists, although is not so common to see [16]. The network consists on a controller and controlled nodes (or slaves), this protocol allows the inclusion of new devices.

A single network can have different Z-Wave chips, we can find a variety of 200, 300 and 400 series. A very important note, chips from 200 and 300 series can't encrypt messages, but the 400 series can do it without problem. For obvious reasons, if you have a network with 400 series, but only one 200 series, the attacker can "sniff" the messages traffic that goes through the WPAN [63].

Thieves can use software like Wireshark (The Wireshark team, California) to observe the behavior of the person living in the house. No need to mention the danger of having some stranger knowing your entire schedule.

Fouladi [28] found an implementation vulnerability in the key exchange protocol that can be used to open the Z-Wave door lock. This attack was done by observing the network traffic for a short time and getting the target device's ID .

One way to countermeasure an attack would be to use only 400 series chips and forward, to make sure that the network can encrypt the messages. Also, the recommendation is to change the Advance Encryption Service (AES) keys in a regular basis [63].

2.3.3 ZigBee

ZigBee is a standard for low-power, low-rate wireless communication used in automation applications [63, 23]. This protocol is based on the IEEE 802.15.4-2003 standard [76].

ZigBee, unlike the other protocols described before, provides facilities for secure communications, like stablish and transport cryptographic keys [23]. The security concepts used by ZigBee are divided in four concepts [76]:

- Security Level: where we can find two security levels (High Security and Standard Security).
- Trust Center: Responsible for the security management. There are three types

of keys (the network key, the master key, and the link key)

- Authentication and Data Encryption: The encryption used is a 128-bit Advanced Encryption Standard (AES) with CCM (Counter with CBC-MAC).

- Integrity and Freshness of Data: ZigBee uses several security keys and methods to make sure there is integrity.

Vidgren [76] describes two possible attacks to this protocol. One of them describes an "End-Device Sabotage Attack", where the attacker impersonates the ZigBee Router (ZR) or the ZigBee Coordinator (ZC) of the ZigBee-enabled system. And the other is a "Network Key Sniffing Attack", where the messages traffic is observed.

Key management is key in this protocol, but if the user doesn't have strong passwords, the protection lowers a lot. In [76], they give some practical countermeasures. For the "End-Device Sabotage Attack", we can use a remote alerting system for warning about power failures, but this requires an active role of a network administrator. And for the "Network Key Sniffing Attack", the user can preinstall the network key when using the Standard Security level, but this is not practical when there is a large network.

2.3.4 1-wire

The 1-wire protocol (Maxim Integrated, California) was designed by Dallas Semiconductor, now known as Maxim Integrated. This protocol is similar to the I2C bus, with the difference that instead of using two cables for transmitting the signal and

other one for power supply [60], 1-wire uses one cable for control, signal transmission and power, and other one as reference [6, 44]. This protocol follows a strict *Master-Slave* scheme, where one or more slave devices can be connected at the same time. Although it is possible, according to Jayapriya's work, where an implementation of 1-wire using multiple masters is shown [37], only a master is enough to fulfill the requirements of most applications where is used [3]. A basic scheme of the protocol is depicted in Figure 2.2.

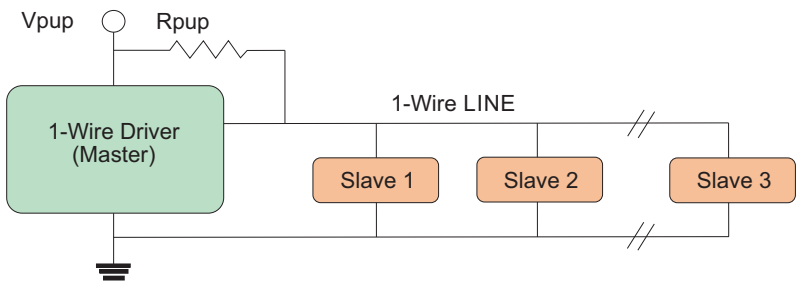

Figure 2.2: Basic scheme of 1-wire

In this protocol, there is no inherent security implemented, there is no encryption or key management at all, it is a simple channel to transmit information. Anyone could connect to the 1-wire bus in any part of the installation and start controlling the system by sending the right commands through the communication line. Also, if the attacker has no knowledge of how the commands are formed, it is possible to observe the behavior of the system. It would be just a matter of time before the intruder guess which commands are used to turn on and off the different devices, or how to control the volume of the music, among other things connected to the bus.

2.4 Lightweight Ciphers

A possibility to implement security in a communication protocol like 1-wire,could be the creation of a cryptographic module before the master and each slave, so we can apply some security to it. This is also to avoid greater costs to the system by making more radical changes. An important factor needs to be addressed in the context of domotics, we need the cryptographic module to fulfill the security and privacy requirements of the users, and at the same time, we need to do it in a way it doesn't interfere with the performance of the system. One of these factors is the response time, we need the information to be encrypted, sent to destination, and decrypted without the user noticing there is something else going on. We need the user's experience to remain unchanged, but with security of their information added to it.

There are a lot of ciphers that could accomplish this requirement. The category of *Lightweight Cryptography* was born to help us with these kind of requirements. With the new arising of the Internet of Things (IoT) paradigm, a new information security problem came along: Ensure information to highly constrained devices in aspects such as memory space, energy, and others. We can find several ciphers that are considered inside this new category such as SIMON & SPECK, Hummingbird, KLEIN, KATAN, TEA, PRESENT, Curupira, DESL, among others [59, 1]. Most of these are block ciphers, which are normally sensitive to linear and differential cryptanalysis. The success of these ciphers lies on the complexity of the algorithm

and the number of rounds used [73]. Depending on the application, we must choose the cipher that best suits our needs.

2.4.1 AES Encryption Algorithm

Rijndael was the winner algorithm of the contest organized by the NIST (National Institute of Standards and Technology) for the new Advanced Encryption Standard (AES) for the United States on the year 2000 [21]. This cipher is a widely used encryption algorithm in many devices, including those categorized into the Internet of Things.

The AES is a block cipher that provides security through a series of operations executed in a determined amount of rounds. Unlike the original version that won the contest, AES only allows the encryption of 128 bits blocks. The allowed sizes for the keys are 128, 192 and 256 bits. Depending on the size of the key, the amount of rounds are 10, 12 and 14 respectively [21, 46, 10].

Each round consists on four transformations, except for the last one:

- ByteSub
- ShiftRow
- MixColumn
- AddRoundKey

For the last round, the same operations are used except for the MixColumn. The decryption process is made by inversing the operations mentioned before [21, 46].

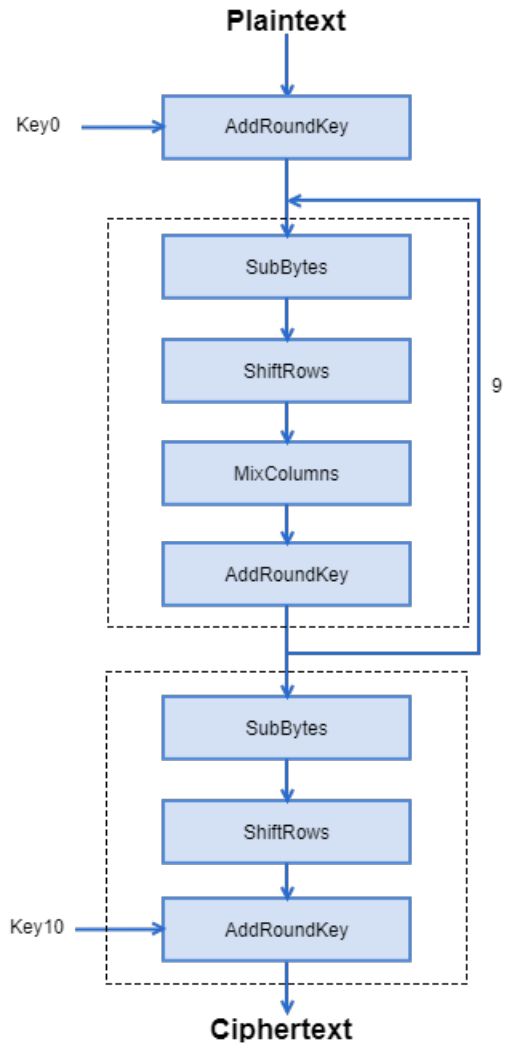

Figure 2.3: Encryption process of the AES algorithm. Based on [46]

In Figure 2.3 and Figure 2.4, we can see the encryption and decryption process of the AES algorithm.

The work of Moradi et. al [52], shows us two successful fault attacks that manage to get the 128-bit key of the AES algorithm. The first one only needs 6 faulty ciphertexts, and the other one around 1500. the fault based cryptanalysis made by Blömer et. al [10] showed that using 256 faulty encryptions was possible to get the

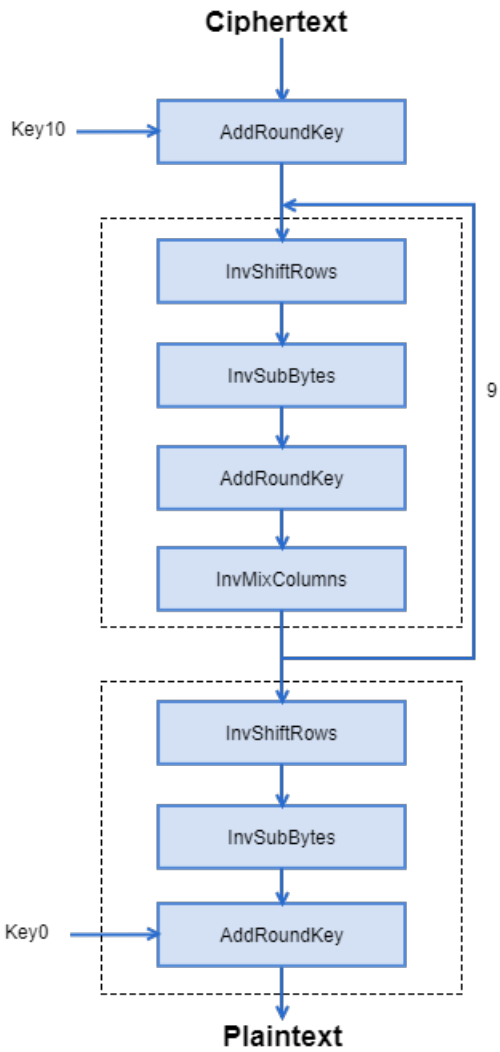

Figure 2.4: Decryption process of the AES algorithm. Based on [46]

key. Kim et. al [40] also tried a differential fault analysis attack, in this work the authors succeeded in getting the key with 4 pairs of correct and faulty ciphertexts in less than 2.3 seconds. Although this algorithm is very fast and useful for many applications, we can see that now it is possible to break the key and get the plaintext.

2.4.2 Hummingbird Lightweight Cipher

Hummingbird is an ultra-lightweight cryptography algorithm created for using on low-cost RFID tags. This particular cipher is a combination of block cipher and stream cipher. The encryption algorithm consists in four 16-bit block ciphers E_{k1}, E_{k2}, E_{k3} and E_{k4}, four 16-bit internal state registers $RS1$, $RS2$, $RS3$ and $RS4$, and a 16-stage LFSR. It uses a 256-bit key that is divided in four 64-bit subkeys called k_1, k_2, k_3 and k_4. Each of these keys are used on the previously mentioned 16-bit block ciphers [24, 25]. To better resume how the encryption and decryption works, see Figure 2.5 and 2.6.

The authors of this cipher claim that Hummingbird is resistant to attacks such as birthday attacks, differential and linear cryptanalysis, structure attacks, algebraic attacks, and cube attacks [25]. Saarinen [66] on the other hand says that there is an exploitable flaw in the initialization process, given that it has a high-bit XOR differential that holds with probability 1. The author also makes an observation that if a collision occurs inside the cipher, there is a one-round iterated differential that works. The attack is successful by attacking K_4, then K_3 and finally K_2. We can see that this cipher can be broken and the information would be revealed. This is an

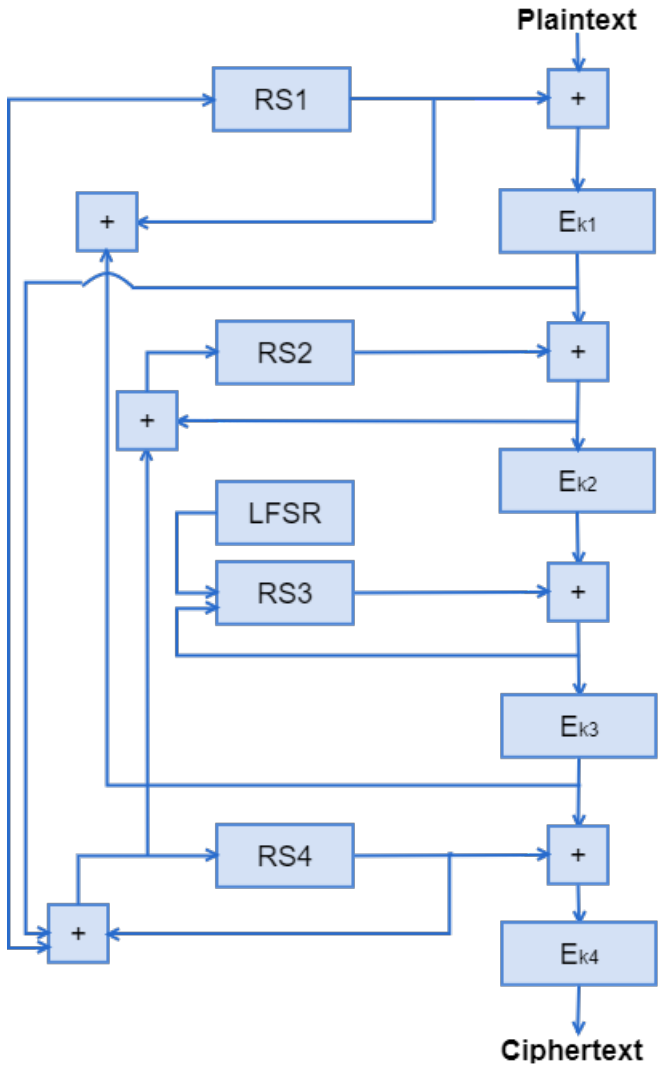

Figure 2.5: Encryption process of the Hummingbird algorithm. Based on [25]

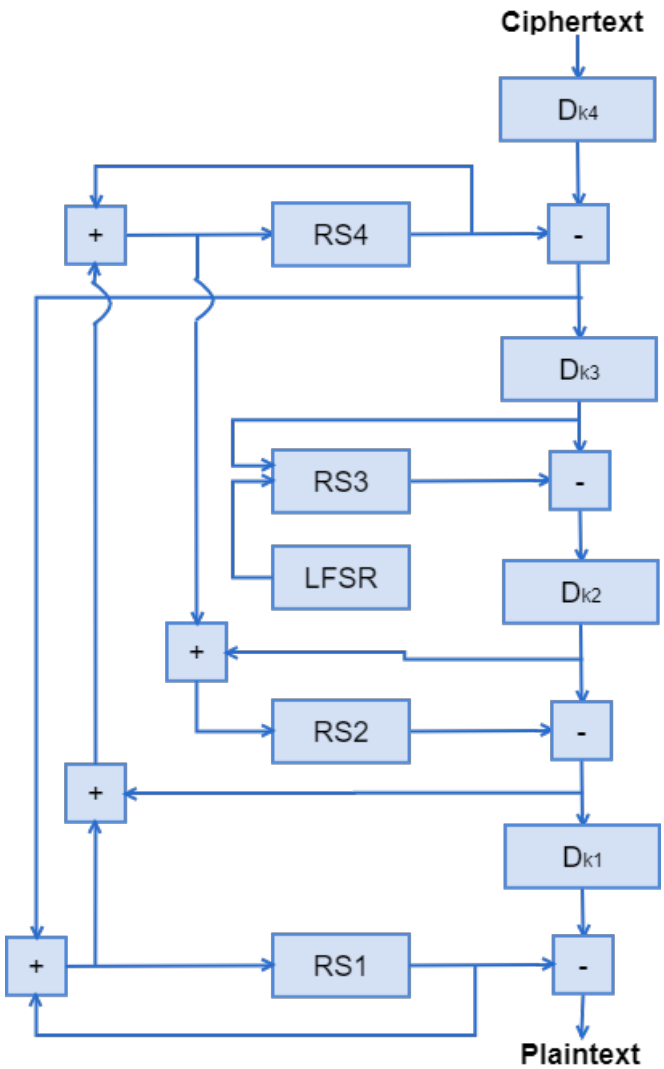

Figure 2.6: Decryption process of the Hummingbird algorithm. Based on [25]

important concern depending on the application. In this case, the major application are RFID cards that can be used in electronic payments, which makes the security a very important issue.

2.4.3 PRESENT

This lighweight cipher was designed specially for very constrained devices like RFID tags. It is an SPN type block cipher and consists on 31 rounds with block sizes of 64 bits. Two key sizes are supported: 80 and 128 bits. Also, this cipher is suitable to be implemented in hardware [11]. The following operations are used in each round:

- AddRoundKey
- sBoxLayer
- pLayer

In Figure 2.7, a description of a PRESENT round is depicted.

Collard et. al [19] claim that the PRESENT cipher was designed with very low security margins. They used a statistical saturation attack and managed to break 15 round of the PRESENT cipher, but they offer the possibility to break more rounds. Another attack was made by Ozen et. al [57], in their work they use a related-key rectangle attack, where two short differential characteristics are used. The authors of this attack claim that it is possible to get the right subkey or at least discard almost all the wrong ones.

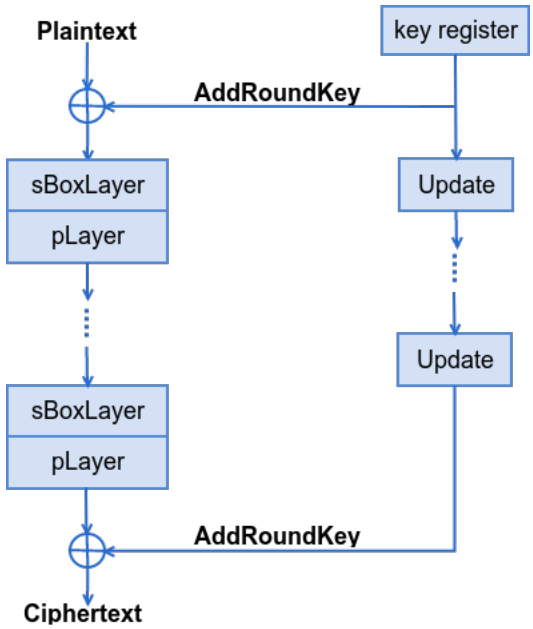

Figure 2.7: One round of PRESENT. Based on [11].

2.4.4 KLEIN Family of Ciphers

KLEIN is a family of lightweight block ciphers that combines a 4-bit Sbox with Rijndael's byte-oriented MixColumn [5]. KLEIN's structure is a Substitution-Permutation Network (SPN), also used by other block ciphers like AES and PRESENT. The algorithm is also built from an involutive 4-bit S-box. The number of round depend on the KLEIN algorithm that is being used. The allowed number of rounds are 12, 16 and 20, for KLEIN-64, 80 and 96 respectively [30].

A KLEIN encryption round follows these steps:

1. AddRoundKey

2. SubNibbles, this operation applies the 4-bit S-box.

3. RotateNibbles

4. MixNibbles, applies two MixColumns in parallel.

The authors of the algorithm claim that this algorithm is efficient in both hardware and software implementations, although it was designed mainly for software. Aumasson et. al [5], applied various attacks to the KLEIN cipher. According to the authors, the attack is possible by a high probability differential. They suggest that the use of a 4-bit Sbox is not a good security feature.

2.4.5 KATAN and KTANTAN Families of Ciphers

KATAN and KTANTAN are two flavors of the same family. Each of them contain three block ciphers. KATAN uses 32, 48 and 64-bit block sizes, and KTANTAN uses the same sizes as well, but it is more compact in hardware because the key is burnt into the device [22].

The KATAN family uses 254 rounds to generate the ciphertext. Two registers L_1 and L_2 are used to load the plaintext. In each round, L_1 and L_2 are shifted to the left. New computed bits are stored in the least significant bits after every shift. After the 254 rounds, you obtain the ciphertext. The only difference between KATAN and KTANTAN is that the second needs a choice of subkey bits to provide some kind of flexibility, given that the key is fixed. The problem is to find a good subkey sequence [22].

In terms of security, Bogdanov et. al [12] describe a Man in the Middle Attack

(MITM) that breaks the KTANTAN family of ciphers. The work of Knellwolf et. al [41] shows a conditional differential cryptanalysis where they recovered the key of the KATAN family of ciphers up to 120 of 254 rounds.

2.4.6 TEA (Tiny Encryption Algorithm)

The Tiny Encryption Algorithm is known to be a very fast block cipher and does not use S-boxes or predefined tables. The rounds follow a Feistel structure and accepts 64-bit blocks and 128-bit keys [77]. The code of is simple and compact, as shown in the following code:

```
void code(long* v, long* w, long* k)
{ unsigned long y=v[0], z=v[1], sum=0,
    delta=0x9e3779b9, n=32;
      while(n-->0)
      {
      sum += delta;
      y+=(z<<4)+k[0]^z+sum^(z>>5)+k[1];
      z+=(y<<4)+k[2]^y+sum^(y>>5)+k[3];
      }
   w[0]=y; w[1]=z;
}
```

Hernández et. al [33] presented a method of constructing distinguishers for cryptographic mapping, proving that the TEA cipher is not secure when using less than five rounds. If they change all operations in the TEA algorithm for XOR, the attack is effective with 128 rounds.

2.4.7 Curupira Block Cipher

Curupira is a block cipher that operates with 96-bits blocks and allows keys of 96, 144 and 192 bits. The number of rounds can vary depending on the key size. For 96-bit keys, the rounds go from 10 to 11. For 144-bit keys, from 14 to 17 rounds. And for 192-bit keys, from 18 to 23 rounds [7].

A round of Curupira can be described by Equation 2.1. Assuming that the subkey numbering starts from k_0 [7].

$$\eta_k(x) = \theta \circ \pi \circ \gamma\, \sigma_k(x) = \theta(\pi(\gamma(\sigma_k(x)))),\ x \in M_n \tag{2.1}$$

Nakahara [54] performed several attacks against the Curupira cipher that were not analyzed by the cipher's authors. This attacks included Impossible-Differential Attack, Boomerang analysis, Plaintext Leakage, Related-key attack and Related-cipher attack. The author explains that is able to attack the Curupira algorithms up to certain rounds, and that the vulnerability depends on the applications where it's used.

2.4.8 DESL

The DESL cipher is a lightweight version of the DES algorithm made to be applied to constrained devices. In this new form of the DES algorithm, the eight S-boxes originally used are replaced for a single one. This algorithm uses a 64-bit block size and a 56-bit key and the ciphertext/plaintext is generated in sixteen rounds [43, 61].

Although this cipher provides better security than the DES cipher, it is still not comparable to the AES algorithm. Also, DESL has a worse performance than the mentioned AES algorithm [62].

2.4.9 Simon and Speck Algorithms

On June 2013, the National Security Agency (NSA) published two families of block ciphers called Simon and Speck [8]. They were created to have a great performance in both hardware and software. Although this is true, the authors make the clarification that Simon shows a better performance in hardware implementations, and Speck in software implementations. The application of these ciphers was not specified, claiming that they are very flexible in terms of their usage. This is due to the range of block and key sizes that are allowed. Simon and Speck have an exceptional performance in 8-bit microcontrollers with minimal flash and SRAM usage, according to the authors.

Each algorithm inside both Simon and Speck are denoted as Simon$2n/mn$ and Speck$2n/mn$, where $2n$ represents the block size in bits and mn the size of the key in bits [8]. Table 2.1 shows the different Simon and Speck versions according to their block size.

Simon Algorithm.

Simon follows a Feistel network structure. As shown in Fig. 2.8, each round requires three main operations:

Table 2.1: Block and key sizes for Simon and Speck. Sizes are in bits.

Block size ($2n$)	**Key size (mn)**
32	64
48	72, 96
64	96, 128
96	96, 144
128	128, 192, 256

- Bitwise XOR, $\bigoplus$
- Bitwise AND, &
- Left circular shift, S^j, by j bits.

One round of Simon is defined by Equation 2.2:

$$R_k(x, y) = (y \oplus f(x) \oplus k, x) \tag{2.2}$$

Where $f(x) = (Sx \& S^8 x) \oplus S^2 x)$, and k is the round key [8].

For decryption, we use the inverse of the previous operation, defined by Equation 2.3:

$$R^{-}1_k(x, y) = (y, x \oplus f(y) \oplus k) \tag{2.3}$$

Simon includes no plaintext and ciphertext whitening steps, because it would affect the size of the circuit. The first and last rounds do nothing cryptographically, they just bring in the first and last round keys [8].

The Simon key schedules use a sequence of 1-bit round constants to eliminate any circular shift symmetries due to the fact that all rounds in Simon are exactly the

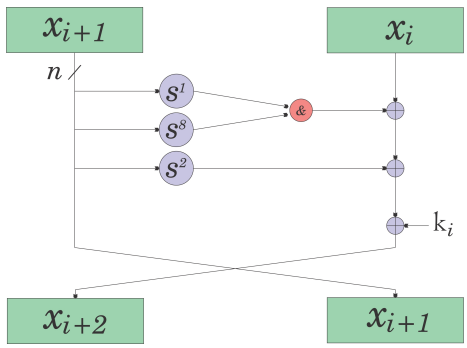

Figure 2.8: One round of Simon. Based on Beaulieu et. al [8]

same apart from the round key. There are five different sequences $(Z_0, ..., Z_4)$ defined to differentiate Simon version with the same block size.

Speck Algorithm.

For a round of SPECK, as shown in Fig. 2.9, we need the following operations:

- Bitwise XOR, $\bigoplus$
- Addition modulo 2^n, + &
- Left and right circular shifts, S^j, S^-j, by j bits.

One round of Speck is defined by Equation 2.4:

$$R_k(x, y) = ((S^{-\alpha}x + y) \oplus k, S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k) \tag{2.4}$$

Where α and β represent the rotation amounts for each round. If $n = 16$, then the values are $\alpha = 7$ and $\beta = 2$. And the same values will be $\alpha = 8$ and $\beta = 3$ for the rest of the values of n.

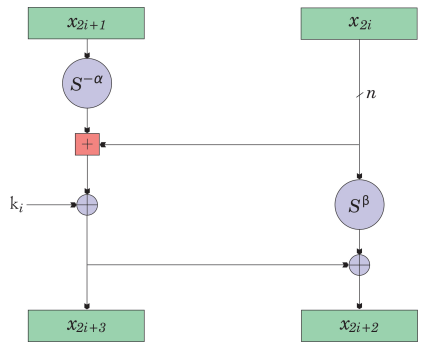

Figure 2.9: One round of Speck. Based on Beaulieu et. al [8]

The inverse function used for decryption is depicted in Equation 2.5:

$$R_k^{-1}(x, y) = ((S^{\alpha}((x \oplus k) - S^{-\beta}(x \oplus y)), S^{-\beta}(x \oplus y)) \tag{2.5}$$

The key schedule of the Speck generates round keys k_i by using the round function. The Key K used is defined by a sequence of values k_i and l_i as $K = (l_{m-2}, ..., l_0, k_0)$. The sequence l_{i+m-1} is defined as $l_{i+m-1} = (k_i + S^{-\alpha} l_i) \oplus i$. The sequence k_i is defined as $k_{i+1} = S^{\beta} k_i \oplus l_{i+m-1}$.

The cryptanalysis of the Simon algorithm made by Alkhzaimi and Lauridsen [2], has shown that in the smaller version of Simon there is a strong differential effect. Tupsamudre et. al [75] have attacked the Simon and Speck families through a fault attack and succeeded

Chapter 3.
State of the Art

Before diving in into how to solve this security issue, we need to make a review of what has already been done around this topic in order to give a working solution. In this work, we are studying a domotic system using 1-wire, but we will also make a revision on how this matter is being handled for other communication protocols (I2C, CAN Bus, SDI-12, SCADA, among others).

3.1 Communication Protocols

Some communication protocols exist that behave in a similar way as 1-wire, allowing a wired channel to transmit data between devices. Some examples of these protocols are I2C, the CAN Bus, SDI-12, SCADA, among others.

For the I2C bus, many algorithms have been used to fulfill the security requirement for each application. Zhu et. al [79], used serial chip with a 64K I2C to transmit data from a radar to another with an acceptable level of confidentiality by using AES. The Atmel SAMA5D3 series of embedded MPU [67], includes hardware engines for encryption with AES, TDES and hash functions to prevent cloning and secure external data transfers. Lázaro et. al [42], presented a way to secure the I2C protocol for

chip-to-chip communications. To secure it, they used the AES-GCM cryptographic and authentication algorithm. The authors claim that the securing this protocol in this way will allow to add security into applications where the computation resources are constrained.

In the context of domotics that use I2C as communication channel, Marginean et. al [47] used Hash based Message Authentication Code (HMAC) to encrypt the information coming from the I2C sensors connected to their system. The authors say that they still need to work with some security experts to be able to know what to do with the data files generated by the system. Mazlan et. al [48], show a smart home network system that supports I2C and SPI communication protocols. They used Programming Lock for Flash Program and EEPROM for data security, but said that this security of the data is not enough when there is a larger amount of nodes.

Chandia et. al [18], made some recommendations for security in SCADA networks, where they suggest to use symmetric keys (AES and SHA1) to guarantee confidentiality and integrity.

Shah et. al [69], created an "Energy conscious home", where the sensors are connected through various communication protocols like Serial UART, I2C, and 1-Wire. The problem is that in this work, the information security is not even a design concern when putting this system together. That is also the case for the work presented by Moraes et. al [53], where they made a hardware implementation for a home automation system using the CAN protocol. The authors even made an

application to control the home remotely, but without considering the implementation of any information security measures.

We can say that for the various wired communication protocols, there is still work to do in terms of securing the information that travels through each line in a system. More importantly, there are still several domotic developments where confidentiality and integrity are not even part of the design. Most of the systems use AES encryption, although it is not entirely safe, as explained in Section 2.4

3.2 1-Wire

The only solution that we found made to address the problem when using 1-wire, it's a device called iButton (Maxim Integrated, California), also developed by Maxim Integrated, used as an authentication token.

The creation of this device was intended to replace many other methods of access control such as smartcards, magnetic stripe cards, barcodes, and radio-frequency proximity cards (RFID). They wanted to provide portability and security to cashless transactions, PKI, authentication, identification, Internet commerce and other processes [68]. For this device, the authors implemented a cryptographic engine that uses SHA-1 (Secure Hash Algorithm 1) to hide the valuable information that stores.

According to the creators of the iButton, this device accomplishes to secure the information contained in it. But as we have found in the literature, there are a couple of ways to break the iButton's security. For instance Kingpin [4], noticed that

a false response created by introducing a false password, it's not entirely random. This response is based on the input and a constant block of data that is being stored inside the iButton. What the author did to find the correct password, was to pre-compute the 48-byte return value that is expected from a wrong answer. Then, the return values are checked to see if they are a match, if the real returned value does not match the pre-computed one, that means the input that was tested it's the correct password.

Another attack that proves the iButton is not completely secure, is a non-invasive side-channel analysis performed by Oswald [56]. The analysis is made to extract the 64 bit secret key that is stored in the device. Also by using a differential fault attack and implementation attack, Brandt et. al [15] managed to get the secret key. This attack is also a very fast one, considering it took them less than ten minutes to prep and execute it.

The iButton still has a long way to be considered as a replacement of smartcards and an option to make electronic payments. Maxim Integrated has other types of solutions to secure the 1-wire protocol, but they all use SHA-1, so we can conclude that they haven't solved this security issue. Also we have to notice that no attempt to secure the 1-wire channel has been done in the context of domotics.

Chapter 4.
Design and Implementation

The approach that we took to provide security for this system required a previous definition of the scenario that we are working on. Then, we proceeded to select a cryptographic algorithm that suits the system, based on the information of various lightweight ciphers described in Section 2.4. By deciding which one to use, we are fulfilling the second specific objective of this work. After that, we made the proper implementation of the selected algorithm to the system through a cryptographic module and that way fulfill the third specific objective.

4.1 Scenario

We chose a domotic system that uses 1-wire as communication protocol as case study for this work. Right now there is no kind of information security implemented. As in some examples in Section 3, security was not considered when designing the system. In its original state, any intruder could connect a master device to the 1-wire bus and start reading and sending commands to the devices as he/she pleases.

Figure 4.1, shows that the original scenario is a Home Area Network (HAN). And the desired scenario is a Wide Area Network (WAN), given that nowadays, the users

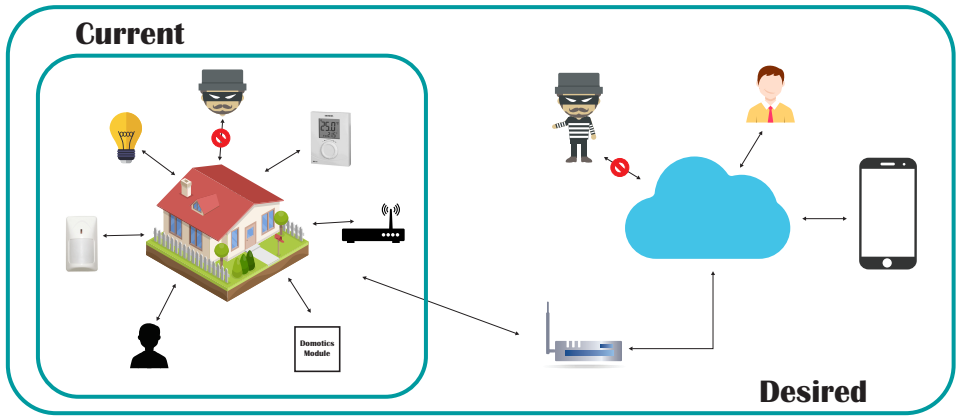

Figure 4.1: General scenario of the system.

have the need to control their system remotely through the Internet. Until we can secure the communication between the devices of the system, we cannot go from HAN to WAN.

4.2 Selection of Cryptographic Algorithm

Looking into the restrictions of the system, mentioned in Section 2.3.4, we concluded that a good strategy to add security is to create a cryptographic module that would go between each module and the 1-wire line. This method also guarantees that the cost of this security upgrade won't be too high to both the domotics company and the final user. By making an analysis of the possible cryptographic algorithms that we could use (Summary of said analysis in Table 4.1), we came to the conclusion of creating the cryptographic module using the Simon and Speck family of ciphers. More specifically, we selected the Speck algorithm, given that it provides a better performance in software implementations.

Table 4.1: Summary of ciphers and their vulnerabilities

Algorithm	**Block Size(bits)**	**Key Size(bits)**	**#Rounds**	**Vulnerability**
AES	128	128, 192, 256	10, 12, 14	Fault Analysis Attack
Hummingbird	Four Blocks of 16-bit	256	N/A	Differential Analysis, Collisions may occur
PRESENT	64	80, 128	31	Statistical Saturation Attack, Related-key Rectangle Attack
KLEIN	4-bit S-box	12, 16, 20, 80, 96	N/A	Differential Analysis
KATAN /KTANTAN	32, 48, 64	80	254	Man in the Middle Attack, Differential Analysis
TEA	64	128	64 Suggested	Cryptographic Mapping
Curupira	96	96, 144, 192	10-23	Impossible-Differential Attack, Boomerang Analysis, Plaintext Leakage, Related-Key Attack, Related-Cipher Attack
DESL	64	56	16	Differential Analysis
Simon/Speck	32, 48, 64, 96, 128	64, 72, 96, 128, 144, 192, 256	22-34	Differential Analysis in the smaller versions

Compared to other Lightweight ciphers, Simon and Speck are said to have an exceptional performance with minimal flash and SRAM usage. They are said to also work better on AVR microcontrollers. Although this algorithm is susceptible to some kind of attacks, it provides good security for our specific application and comes with a great performance on the implementation of the cryptographic module.

4.3 Implementation

In this section we are going to explain what did we take into account when designing this custom solution for our case study. The hardware used were a Raspberry Pi 3 model B (The Raspberry Pi Foundation, UK) as a master that can connect to the Internet, and an Arduino Uno (Arduino, Italy) as the cryptographic module. The software that we used included Python 2.7 (Python 2.7.0 Release, 2010) and the Arduino IDE (Arduino, Italy) with the necessary libraries.

4.3.1 Hardware

As we said before, users nowadays want to control their domotic system remotely from anywhere. To make this possible for the system, a Raspberry Pi 3. was included to the system. The Raspberry Pi 3 is added as a master of the system, with the capability of sending commands on its own. By adding this device, we could connect to the system and start observing what was going on with the whole system. Proving that way that it is possible to know what the different commands do without too much effort.

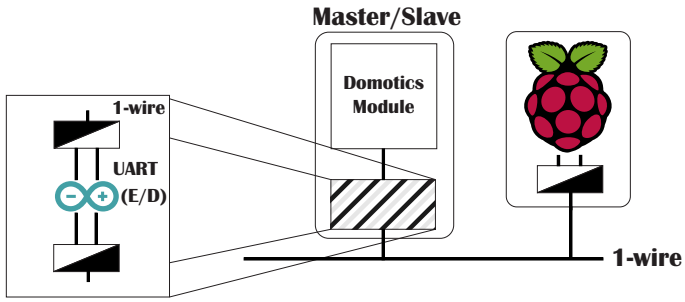

Figure 4.2: Implementation on the domotic system.

To make the cryptographic module, we used an Arduino Uno as depicted in the left-hand side of Figure 4.2. Also to be able to communicate the Arduino Uno with the 1-wire protocol, we added an interface that allows this communication between the serial port of the Arduino Uno (Tx and Rx pins, also known as UART) and the 1-wire channel. The Arduino Uno, as cryptographic module, is in charge of encrypting and decrypting the system's commands and send the results where it needs to go. The intention of using the Arduino Uno, is to be able to make a prototype of the cryptographic module, and then move to an integrated chip that will do the same work, but in a much smaller space. That way it will be possible to include it inside the domotics module.

The characteristics of both Raspberry Pi and Arduino are shown in Table 4.3 and Table respectively.

Table 4.2: Parameters of the Raspberry Pi 3 used in this work.

Parameter	**Value**
CPU Model	64-bit quad-core ARMv8
Frequency	1.2 GHz
RAM	1GB
Operating System	Raspbian Jessie

Table 4.3: Parameters of the Arduino Uno used in this work.

Parameter	Value
Microcontroller	ATmega328P
Clock speed	16 MHz
SRAM	2 Kb
Flash memory	32 Kb
EEPROM	1 Kb

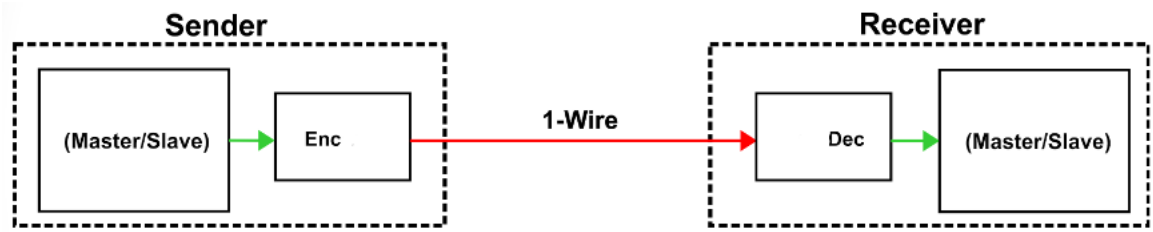

Figure 4.3: Block diagram showing the connection of the system.

In Figure 4.3, we show how the cryptographic module is connected to the domotic system. The blocks showing *Master/Slave* in the figure, represent the original domotic modules without any modification. The blocks next to the domotic modules, represent the cryptographic module in charge of encrypting or decrypting the incoming message. When joining the original and the cryptographic modules, we make sure that no one can be connected to the system between these two modules.

4.3.2 Software

For the two platforms that we are managing in this system (Raspberry Pi 3 and Arduino Uno), we used a Python implementation of the algorithm and an Arduino library respectively.

Raspberry Pi

The Raspberry Pi that was implemented as a master, it does not need to have one of the cryptographic modules mentioned above, given that it can encrypt and send, and receive and decrypt on its own.

The implementation of the cryptographic algorithm was based on the Simon and Speck python algorithms by Calvin McCoy [49]. The proprietary protocol of this system we are studying, use different sizes of commands depending on the function. Because of this, and that the cryptographic algorithms need a specific block size to be defined in order to make the encryption and decryption, we divide the commands in chunks of the desired block size. Then, we encrypt each part separately to join all of them at the end of the process and send the result. We follow the same process when decrypting the commands. This provides flexibility in terms of the size of the commands sent by the system. We could have a very large command and encrypt and decrypt it successfully. If a command is shorter than the defined block size, it would not affect the process, given that as described in the Simon and Speck algorithms [8], the remaining space would be filled with zeros and follow the encryption normally.

To test the functionality of the encryption, we measured the time it takes to the Raspberry Pi to encrypt different lengths of commands using the Simon and Speck algorithm, as seen in the published work based on these results [20]. In this case, we divided the different commands in 32 bit chunks and proceeded with the encryption process using the Simon32/64 and Speck32/64 algorithms. Then we compared the

encryption/decryption times with two objectives: Check if the extra time taken to send each command due to the encryption/decryption process would affect the user's experience, and prove which of both algorithms would suit better for our application.

Arduino UNO

To implement the same process on the Arduino Uno, we used the *Crypto* library created by Github user rweather [65]. In the author's documentation [65], it is said that the use of the families with block sizes of 32, 48, 64, or 96 bits are too small to be used in modern cryptosystems. That's why the recommendation is to use the 128-bit block size with 128-bit, 192-bit, or 256-bit key sizes.

In Figure 4.4, we can see a flow diagram on how the Arduino software works. For the encryption and decryption part (ENC/DEC), we used the function provided by the Speck library. In Figure 4.5, we can see the flowchart describing that process.

In summary, the Arduino checks if there is information available on the serial ports, if there is, takes the string and depending from which port comes from, the string gets encrypted or decrypted. The last step is to send the encrypted/decrypted string through the opposite serial port.

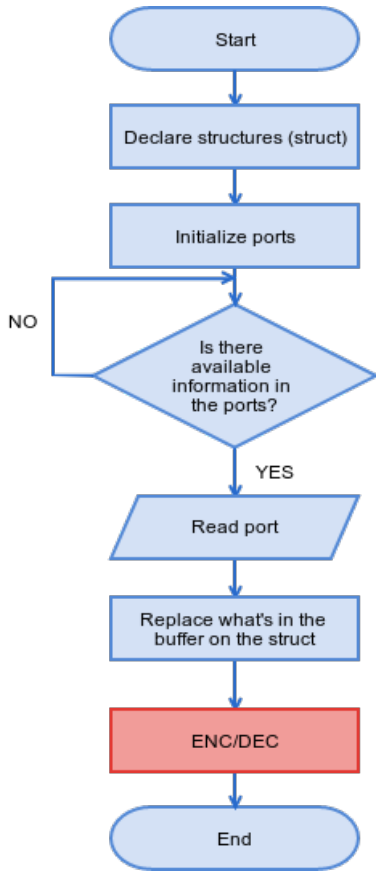

Figure 4.4: Flowchart describing the process followed by the Arduino program.

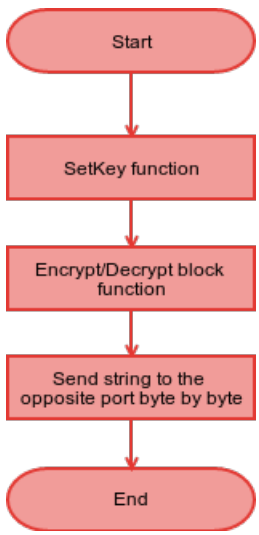

Figure 4.5: Flowchart describing the process followed by the Arduino program in the encryption part.

Chapter 5.
Results

The implementation of the Arduino Uno to the system is shown in Figure 5.1. We are capable of communicate all the devices included in the implementation. The Arduino Uno is capable of encrypting the commands that it receives and send them through the 1-wire channel. In the same way, it is capable of receiving an incoming command and decrypting it to send it to the domotics module. We also have tested how long it takes for the whole process to happen, with the encrypting/decryption added to the system. This way, besides guaranteeing security in the domotic system, we can also see if this change would be perceptible to the users when using the system. The results shown in Section 5.1 and 5.2, are included in the paper published based on this work [20].

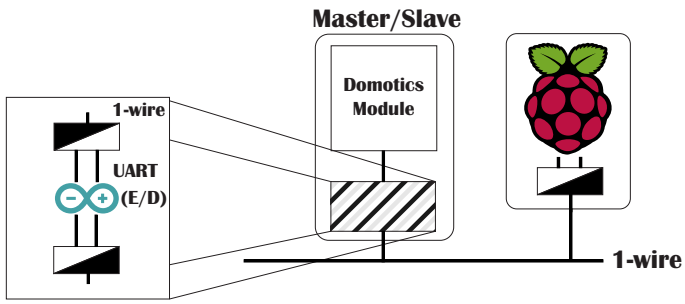

Figure 5.1: Implementation on the domotic system.

Table 5.1: Execution time data

	Algorithm			
Length	**Speck [ms]**		**Simon [ms]**	
32 bits	0.5411137	0.5409502	0.8170261	0.8228535
48 bits	0.8173872	0.8144342	1.452703	1.453392
64 bits	0.813048	0.8173257	1.453032	1.457409
96 bits	1.099465	1.093212	2.10735	2.10833
128 bits	1.381253	1.376364	2.719511	2.721607
160 bits	1.639491	1.624422	3.356103	3.382519
192 bits	1.932254	1.915469	3.980706	3.986703

5.1 Execution Time

The execution time tells us how much time the algorithm takes to encrypt a command with a determined length. Because this algorithm is symmetric, we assume that the decryption time is almost the same as the encryption.

To prove which of the algorithms (Simon and Speck) is the best for our system, we used a mixed factorial design to check if the length of the commands would affect the response time of the system at the eyes of the final user. We used a Raspberry Pi 3 model B to execute the encryption algorithms and measure the execution times.

The lengths of the commands tested were 32, 48, 64, 96, 128, 160 and 192 bits. For the measures times to be statistically significant, we needed to run the encryption program several times. We ran the program 10,000 times with one replicate to make and then take the average of those measures. The *R* software (The R Foundation, Austria) was used to obtain the graphics where the data is shown. In Table 5.1, we can see the measured execution times for both Speck and Simon algorithm and the different lengths of commands.

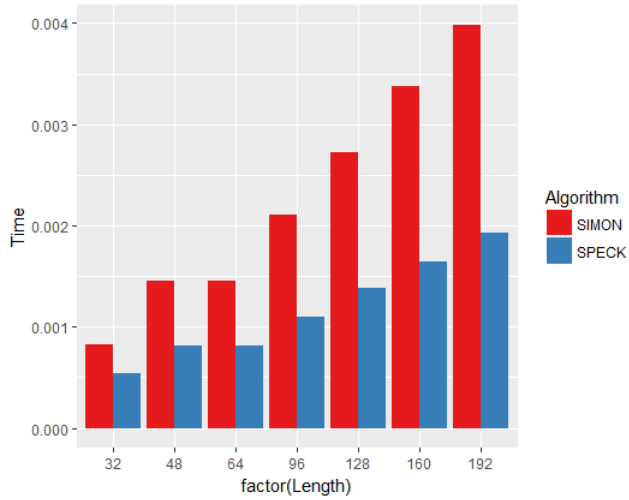

Figure 5.2: Comparison of execution times between Simon and Speck algorithms. The time is in seconds and the length is in bits.

We can see the difference in execution time on Figure 5.2. There is a clear difference between the two algorithms when seen in the graph. The execution time on the Speck algorithm does not increase by a lot when changing the commands length. Simon on the other hand, has a greater difference when encrypting a 32 bits command and a 192 bits one. Just by looking at the figure, the obvious decision is to go with the Speck algorithm. We should also notice that the time is shown in seconds, so in reality, the difference between the algorithms it's not too much. Simon only takes 0.002 s more than Speck when encrypting the largest command. But still, we need to make sure that these times, in both algorithms would not affect the response time of the system.

5.2 Latency of the System

The importance of looking into the latency of the system relies on the fact that we are checking if the cryptographic module would affect the user's experience when using the domotic system.

The baud rate of the 1-wire protocol is 9600 baud. We can calculate the time taken for the protocol to send one bit by using Equation 5.1.

$$bittime = \frac{1}{9600} = 104\mu s \tag{5.1}$$

The 1-wire protocol takes $104\mu s$ to send one bit of information. Having this information, we can calculate that for a 32 bit string, the system would take $3.328ms$ to send it. And for the largest command that we tested previously, it would take $19.968ms$.

To understand better how these times of encryption and transmission affect the performance of the system, we can see at Figure 5.3, where the process from sending the command from one module to the reception in the final destination is depicted. The command is first transmitted to the cryptographic module, the command gets encrypted. Then is transmitted again until it gets to the next cryptographic module, in which the command is decrypted. Finally, the command is transmitted once more to get to the module that receives the order.

Looking at the data in Table 5.1, we can take for example, the average times

Figure 5.3: Process to send a command.

when encrypting the largest command available in the system (192 bits). For Speck, this time 1.9 milliseconds, and for Simon y 3.9 milliseconds. Taking into account the process mentioned above, we have that the whole process to send a 192 bits command takes $63.704ms$ and $67.704ms$ using Simon and Speck respectively. When making this calculation, we can clearly see that the difference between both algorithms is not much, given that the command still needs to go through the transmission in the 1-wire line three times. According to Miller [50], anything that occurs in less than 100 ms is perceived as instantaneous by an average human being. Therefore, the whole process of sending, encrypting/decrypting would not be noticeable by the users. The latency of the system would be up to $63.704ms$ or $67.704ms$ depending on which algorithm is used. Shorter commands obviously would take less time than that. Although the latency is bellow 100ms, we still need to find a way to reduce these times even more.

5.3 Security

The main objective of this work is the implementation of security within the domotic system. We have implemented the Speck algorithm in the cryptographic module using an Arduino Uno. Given the cryptoanalysis described in Section 2.4.9, we could not use Simon or Speck on their smaller versions. The smaller versions would

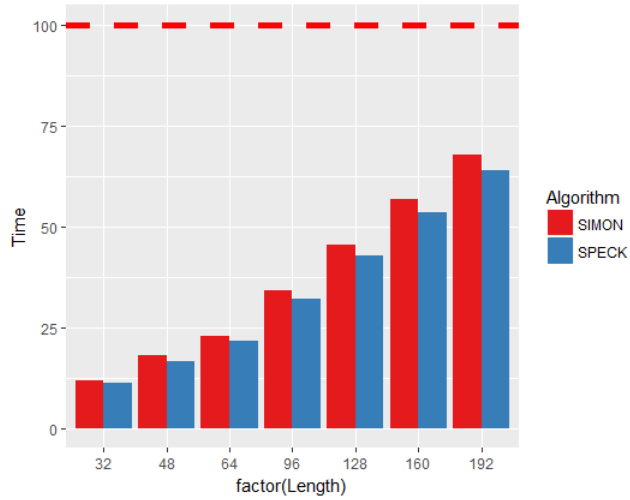

Figure 5.4: Comparison of execution times between Simon and Speck algorithms. The time is in seconds and the length is in bits.

make the system vulnerable to differential attacks easier. This is why we used the Speck128/128 version.

The first test that we made was to verify that the results matched the test vectors provided by the authors of the algorithms [8]. Then we tested the implementation of the algorithm within the code used to communicate the Arduino Uno, the Raspberry Pi and the domotic system. Here, we could successfully send a command from the Raspberry pi and accomplish the sending process described in Figure 5.3. Then, with the Raspberry connected to see what's happening on the system, as an intruder would do, we performed any action on the domotic system (turn on/off, change the channel, play next song, etc) and watched how the response received by the Raspberry was unintelligible and impossible to relate to an specific order of commands.

The security in this kind of implementation is given by the security of the algo-

rithm that is used on the microcontroller. This gives us a great flexibility in terms of choosing a cipher that better suits the application. We could use this same methodology not only with the 1-wire protocol, but with other communication protocols that act in a similar way.

Chapter 6.
Discussion and Future work

In this work we accomplished the main objective of this work by providing security to the 1-wire channel. Although the chosen ciphers are not entirely secure against differential attacks, it gives enough security so an intruder that gets to the system cannot understand or find a way to know how the commands are made. It is important to note that it's not just about the applied cipher, there is also a need of speed and space on this kind of implementations. All of it to give the users a better experience with the system.

Because we are never completely secure against new attacks to the ciphers or the systems, as future work we would still need to study further security measures to avoid any attacks to the system. Here we have focused on the security triad (Confidentiality, Integrity and Availability), but we could start searching for a way to guarantee authentication, access control and no-repudiation as well. It would also be interesting to test this methodology on other communication protocols and ciphers.

Chapter 7.
Conclusions and Recommendations

We have implemented a custom solution for the domotic system studied, that uses 1-wire as communication protocol. We have fulfilled the objectives of this thesis work by identifying the restrictions of our system, studying the different cryptographic algorithms that could suit the system and choose one of them. Finally, we have made an implementation to the system where it is possible to send all the commands encrypted for no intruder to understand or use against the owner of the system. Given that there are new ciphers everyday and new ways to break all of the existing ones, we need to keep an eye on these changes so we can apply them to our now secured domotic system.

We have published a scientific article from the preliminary results of this work (Sections 5.1 and 5.2) that shows if the cryptographic module would affect the final user's experience [20].

We can conclude that we have developed a flexible methodology to add security to a wired domotic system while preserving a good performance and keeping the costs low for the manufacturer of the system.

Bibliography

[1] M. Alizadeh, M. Salleh, M. Zamani, J. Shayan, and S. Karamizadeh. Security and performance evaluation of lightweight cryptographic algorithms in rfid. *Kos Island, Greece*, 2012.

[2] H. AlKhzaimi and M. M. Lauridsen. Cryptanalysis of the simon family of block ciphers. *IACR Cryptology ePrint Archive*, 2013:543, 2013.

[3] ATMEL Corporation. AVR318: Dallas 1-wire master. Technical report, 2004.

[4] K. atstake. Ds1991 multikey iButton dictionary attack vulnerability, 2001.

[5] J.-P. Aumasson, M. Naya-Plasencia, and M.-J. O. Saarinen. Practical attack on 8 rounds of the lightweight block cipher klein. In *INDOCRYPT*, volume 7107, pages 134–145. Springer, 2011.

[6] D. Awtrey and D. Semiconductor. Transmitting data and power over a one-wire bus. *Sensors-The Journal of Applied Sensing Technology*, 14(2):48–51, 1997.

[7] P. Barreto and M. Simplicio. Curupira, a block cipher for constrained platforms. *Anais do 25o Simpsio Brasileiro de Redes de Computadores e Sistemas Distribudos-SBRC*, 1:61–74, 2007.

[8] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The simon and speck lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference*, page 175. ACM, 2015.

[9] R. Bertuzzi, P. Guarda, and J. P. Salazar. Automatización del hogar usando el protocolo de comunicación x10.

[10] J. Blömer and J.-P. Seifert. Fault based cryptanalysis of the advanced encryption standard (aes). In *Computer Aided Verification*, pages 162–181. Springer, 2003.

[11] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In *CHES*, volume 4727, pages 450–466. Springer, 2007.

[12] A. Bogdanov and C. Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher ktantan. In *Selected Areas in Cryptography*, volume 6544, pages 229–240. Springer, 2010.

[13] H. Bouma. Gerontechnology: Making technology relevant for the elderly. *Gerontechnology*, 3:1, 1992.

[14] H. Bouma, J. L. Fozard, D. G. Bouwhuis, and V. Taipale. Gerontechnology in perspective. *Gerontechnology*, 6(4):190–216, 2007.

[15] C. Brandt and M. Kasper. Don't push it: Breaking ibutton security. In *Foundations and Practice of Security*, pages 369–387. Springer, 2014.

[16] D. C. Brewer. *Home automation made easy: do it yourself know how using UPB, Insteon, X10 and Z-Wave*. Que Publishing, 2013.

[17] A. Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, and C. Dixon. Home automation in the wild: challenges and opportunities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2115–2124. ACM, 2011.

[18] R. Chandia, J. Gonzalez, T. Kilpatrick, M. Papa, and S. Shenoi. Security strategies for scada networks. *Critical Infrastructure Protection*, pages 117–131, 2007.

[19] B. Collard and F.-X. Standaert. A statistical saturation attack against the block cipher present. In *CT-RSA*, volume 5473, pages 195–210. Springer, 2009.

[20] L. A. M. Colorado and J. C. Martíinez-Santos. Leveraging 1-wire communication bus system for secure home automation. In *Colombian Conference on Computing*, pages 759–771. Springer, 2017.

[21] J. Daemen and V. Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[22] C. De Canniere, O. Dunkelman, and M. Knežević. Katan and ktantan—a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 272–288. Springer, 2009.

[23] G. Dini and M. Tiloca. Considerations on security in zigbee networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, pages 58–65. IEEE, 2010.

[24] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith. Ultra-lightweight cryptography for low-cost rfid tags: Hummingbird algorithm and protocol. *Centre for Applied Cryptographic Research (CACR) Technical Reports*, 29, 2009.

[25] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith. Hummingbird: ultra-lightweight cryptography for resource-constrained devices. In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2010.

[26] EuroX10. ¿Qué es el X-10?, 2015.

[27] M. Farooq, M. Waseem, A. Khairi, and S. Mazhar. A critical analysis on the security concerns of internet of things (iot). *International Journal of Computer Applications*, 111(7), 2015.

[28] B. Fouladi and S. Ghanoun. Security evaluation of the z-wave wireless protocol. *Black hat USA*, 24, 2013.

[29] J. Fozard. Gerontechnology and perceptual motor-function: New opportunities for prevention, compensation, and enhancement. *Gerontechnology*, 1(1):5–24, 2001.

[30] Z. Gong, S. Nikova, and Y. W. Law. Klein: A new family of lightweight block ciphers. *RFIDSec*, 7055:1–18, 2011.

[31] S. Goodwin. *Smart home automation with Linux and Raspberry Pi*. Apress, 2013.

[32] D. Hendricks. The history of smart homes, April 2014.

[33] J. C. Hernandez and P. Isasi. Finding efficient distinguishers for cryptographic mappings, with an application to the block cipher tea. *Computational Intelligence*, 20(3):517–525, 2004.

[34] K. J. Higgins. Popular home automation system backdoored via unpatched flaw, April 2015.

[35] K. Hill. When 'smart homes' get hacked: I haunted a complete stranger's house via the internet, 2013.

[36] B. Hodge, E. Joshi, S. Khandelwal, and Y. Kothari. Android based home automation using raspberry pi. *Imperial Journal of Interdisciplinary Research*, 2(5), 2016.

[37] M. Jayapriya, T. Rajvignesh, and A. Nandini. Single wire common protocol for multiple masters. 2015.

[38] M. E. Jima Masache and J. H. Zambrano Espinosa. *Diseño e implementación de un prototipo de sistema domótico para permitir el control centralizado de dispositivos dentro del hogar controlado por FPGA utilizando el protocolo X10*. PhD thesis, Quito: EPN, 2015., 2015.

[39] D. Kennedy and R. Simon. Pentesting social-engineering over power lines, 2011.

[40] C. H. Kim and J.-J. Quisquater. New differential fault analysis on aes key schedule: Two faults are enough. In *CARDIS*, volume 5189, pages 48–60. Springer, 2008.

[41] S. Knellwolf, W. Meier, and M. Naya-Plasencia. Conditional differential cryptanalysis of trivium and katan. In *Selected Areas in Cryptography*, volume 7118, pages 200–212. Springer, 2011.

[42] J. Lázaro, A. Astarloa, A. Zuloaga, U. Bidarte, and J. Jiménez. I2csec: A secure serial chip-to-chip communication protocol. *Journal of Systems Architecture*, 57(2):206–213, 2011.

[43] G. Leander, C. Paar, A. Poschmann, and K. Schramm. New lightweight des variants. In *International Workshop on Fast Software Encryption*, pages 196–210. Springer, 2007.

[44] B. Linke. Overview of 1-wire® technology and its use. *MAXIM, AN1796, jun*, 2008.

[45] S. Lucero and K. Burden. Home automation and control. *ABI Research*, 2010.

[46] P. Mahajan and A. Sachdeva. A study of encryption algorithms aes, des and rsa for security. *Global Journal of Computer Science and Technology*, 2013.

[47] M.-T. Marginean and C. Lu. sdomo—a simple communication protocol for home automation and robotic systems. In *Technologies for Practical Robot Applications (TePRA), 2015 IEEE International Conference on*, pages 1–7. IEEE, 2015.

[48] M. H. Mazlan, F. Mohamad, and R. A. Rashid. Smart home networking system.

[49] C. McCoy. Implementations of the simon and speck block ciphers, March 2015.

[50] R. B. Miller. Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 267–277. ACM, 1968.

[51] N. C. Moore. Hacking into homes: 'smart home' security flaws found in popular system, May 2016.

[52] A. Moradi, M. Shalmani, and M. Salmasizadeh. A generalized method of differential fault attack against aes cryptosystem. *Cryptographic Hardware and Embedded Systems-CHES 2006*, pages 91–100, 2006.

[53] F. Moraes, A. Amory, N. Calazans, E. Bezerra, and J. Petrini. Using the can protocol and reconfigurable computing technology for web-based smart house automation. In *Integrated Circuits and Systems Design, 2001, 14th Symposium on.*, pages 38–43. IEEE, 2001.

[54] J. Nakahara. Analysis of curupira block cipher. *Anais do 8o Simpsio Brasileiro em Segurana da Informao e Sistemas Computacionais*, 2008.

[55] T. Oluwafemi, T. Kohno, S. Gupta, and S. Patel. Experimental security analyses of non-networked compact fluorescent lamps: A case study of home automation security. In *Proceedings of the LASER 2013 (LASER 2013)*, pages 13–24, 2013.

[56] D. Oswald. Side-channel attacks on sha-1-based product authentication ics. In *International Conference on Smart Card Research and Advanced Applications*, pages 3–14. Springer, 2015.

[57] O. Özen, K. Varıcı, C. Tezcan, and Ç. Kocair. Lightweight block ciphers revisited: Cryptanalysis of reduced round present and hight. In *Information security and privacy*, pages 90–107. Springer, 2009.

[58] P. Paganini. How hackers violate privacy and security of the smart home, September 2015.

[59] S. Panasenko and S. Smagin. Lightweight cryptography: Underlying principles and approaches. *International Journal of Computer Theory and Engineering*, 3(4):516, 2011.

[60] Philips Semiconductors. The I2C-bus specification. *Philips Semiconductors*, 9397(750):00954, 2000.

[61] A. Poschmann, G. Leander, K. Schramm, and C. Paar. A family of light-weight block ciphers based on des suited for rfid applications. In *Workshop on RFID Security–RFIDSec*, volume 6, 2006.

[62] S. Rinne, T. Eisenbarth, and C. Paar. Performance analysis of contemporary light-weight block ciphers on 8-bit microcontrollers. In *ECRYPT Workshop SPEED-Software Performance Enhancement for Encryption and Decryption, Amsterdam*, 2007.

[63] R. Romera. Home automation and cybercrime. Technical report, 2013.

[64] L. Rothfeld. Tech time machine: The smart home, January 2015.

[65] rweather. Arduinolibs - cryptographic library, August 2016.

[66] M.-J. O. Saarinen. Cryptanalysis of hummingbird-1. In *FSE*, volume 11, pages 328–341. Springer, 2011.

[67] SAMA5D3 Series. ARM-based Embedded MPU.

[68] D. Semiconductor iButton. A practical introduction to the dallas semiconductor ibutton^{TM}.

[69] V. Shah, D. Chow, S. Pinsky, D. Serenelli, and B. Skolozdra. Energy-concious home automation.

[70] R. W. Shirey. Internet security glossary, version 2. 2007.

[71] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers. Twenty security considerations for cloud-supported internet of things. 2016.

[72] D. Spicer. The echo iv home computer: 50 years later, May 2016.

[73] M. Stamp. *Information security: principles and practice*. John Wiley & Sons, 2011.

[74] Statista. Digital market outlook: Smart home, 2016.

[75] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay. Differential fault analysis on the families of simon and speck ciphers. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 40–48. IEEE, 2014.

[76] N. Vidgren, K. Haataja, J. L. Patino-Andres, J. J. Ramirez-Sanchis, and P. Toivanen. Security threats in zigbee-enabled systems: vulnerability evaluation, practical experiments, countermeasures, and lessons learned. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 5132–5138. IEEE, 2013.

[77] D. J. Wheeler and R. M. Needham. Tea, a tiny encryption algorithm. In *International Workshop on Fast Software Encryption*, pages 363–366. Springer, 1994.

[78] G. Wiszniewski. History and invention of the vacuum cleaner, 2017.

[79] X. Zhu, W. Yu, and G. Jin. An encryption technique for measurement and display of radar parameters. 2015.